

FIG. 1

(a)

```
int a[];  
for (i=0; i<128; i++){  
    x += a[i];  
}
```

⋮

(b)

```
int a[];  
for (i=0; i<128; i++){  
    dpref(&a[i + N]);  
    x += a[i];  
}
```

⋮

// Prefetch data a certain number (N, in this case) of iterations ahead
// in consideration of latency caused until reference is made

FIG. 2

```
for (i = 0; i < 128; ) {  
    dpref(&a[i+32]);  
    for (j = 0; j < 32; j++, i++) {  
        x += a[i];  
    }  
}
```

⋮

FIG. 3

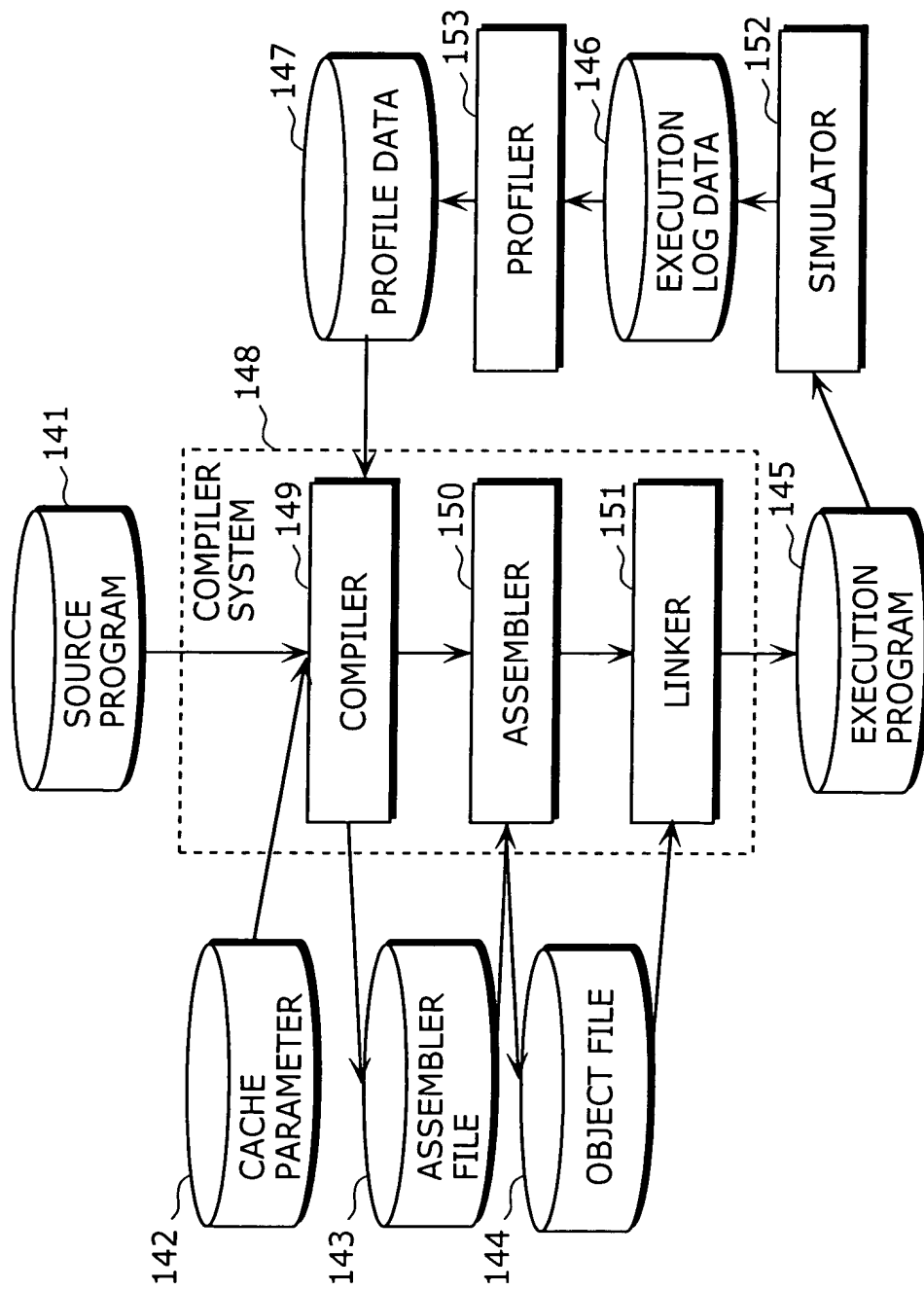


FIG. 4

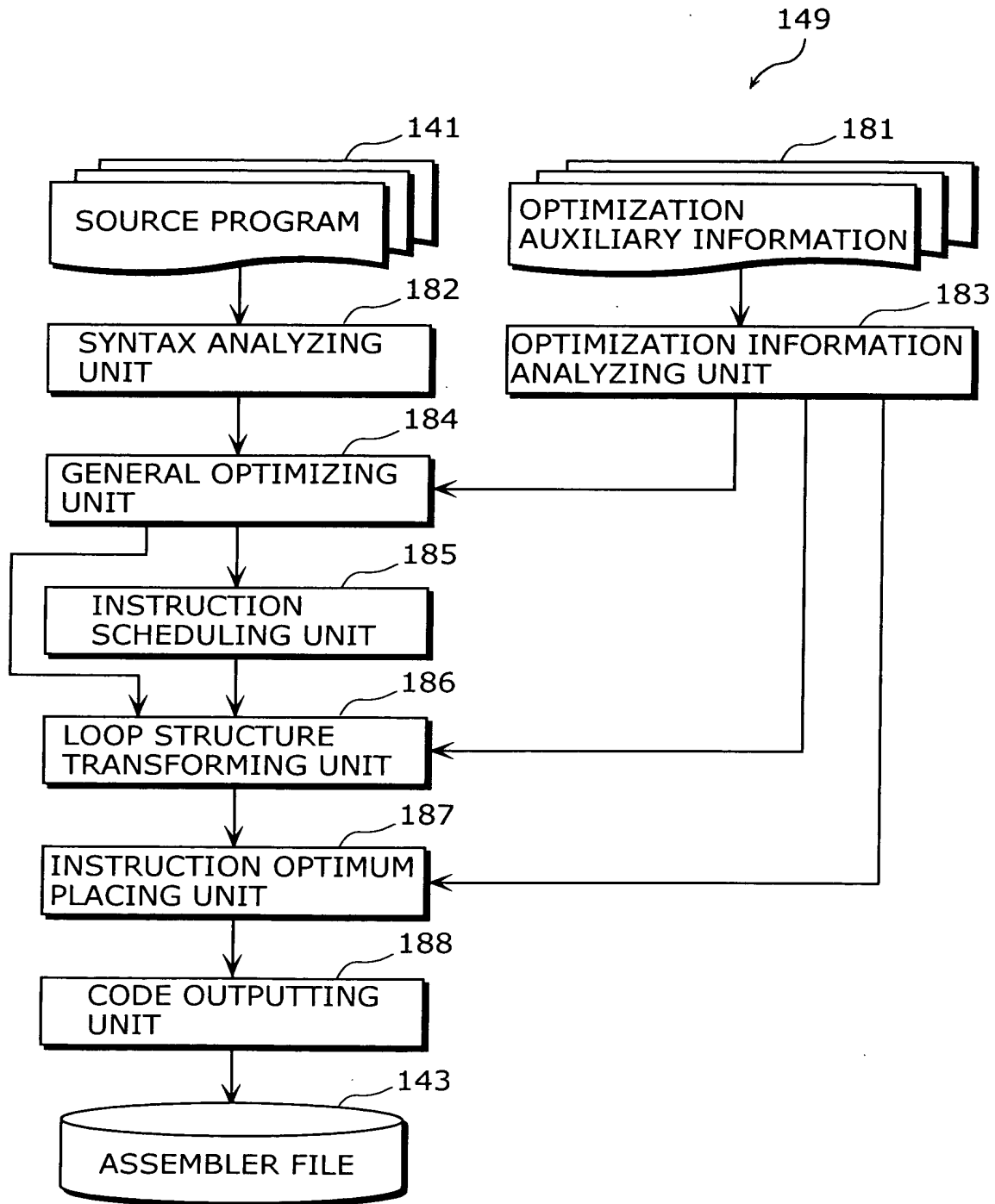


FIG. 5

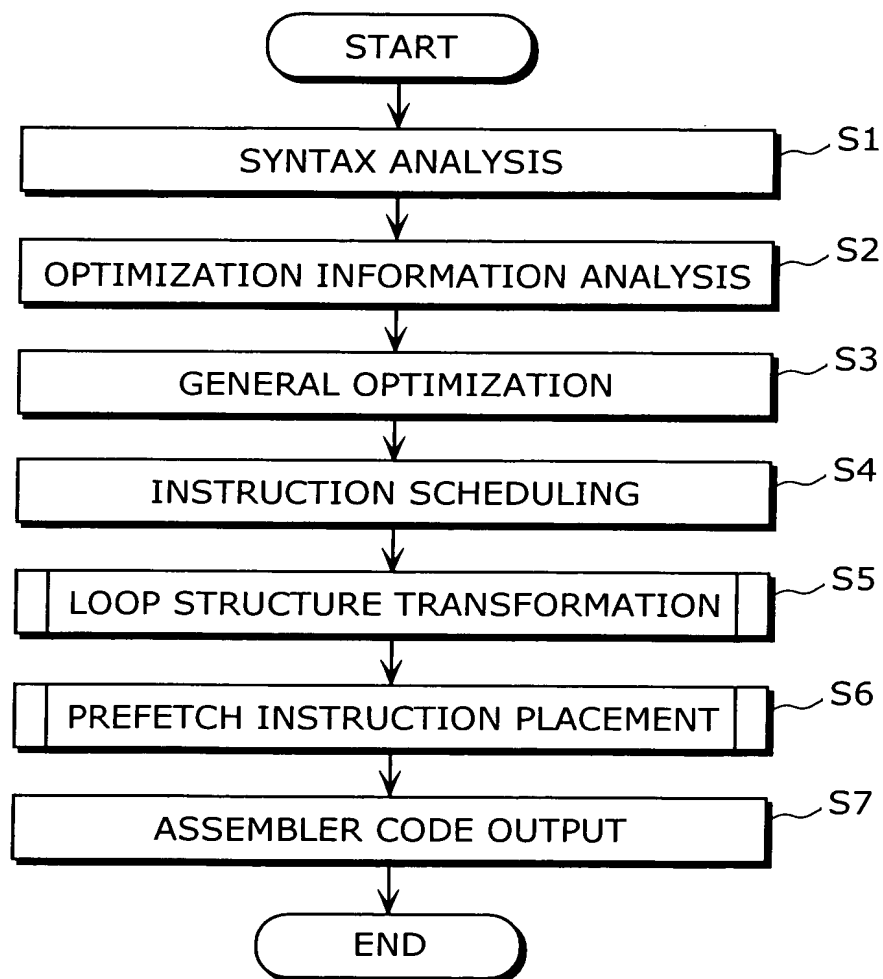


FIG. 6

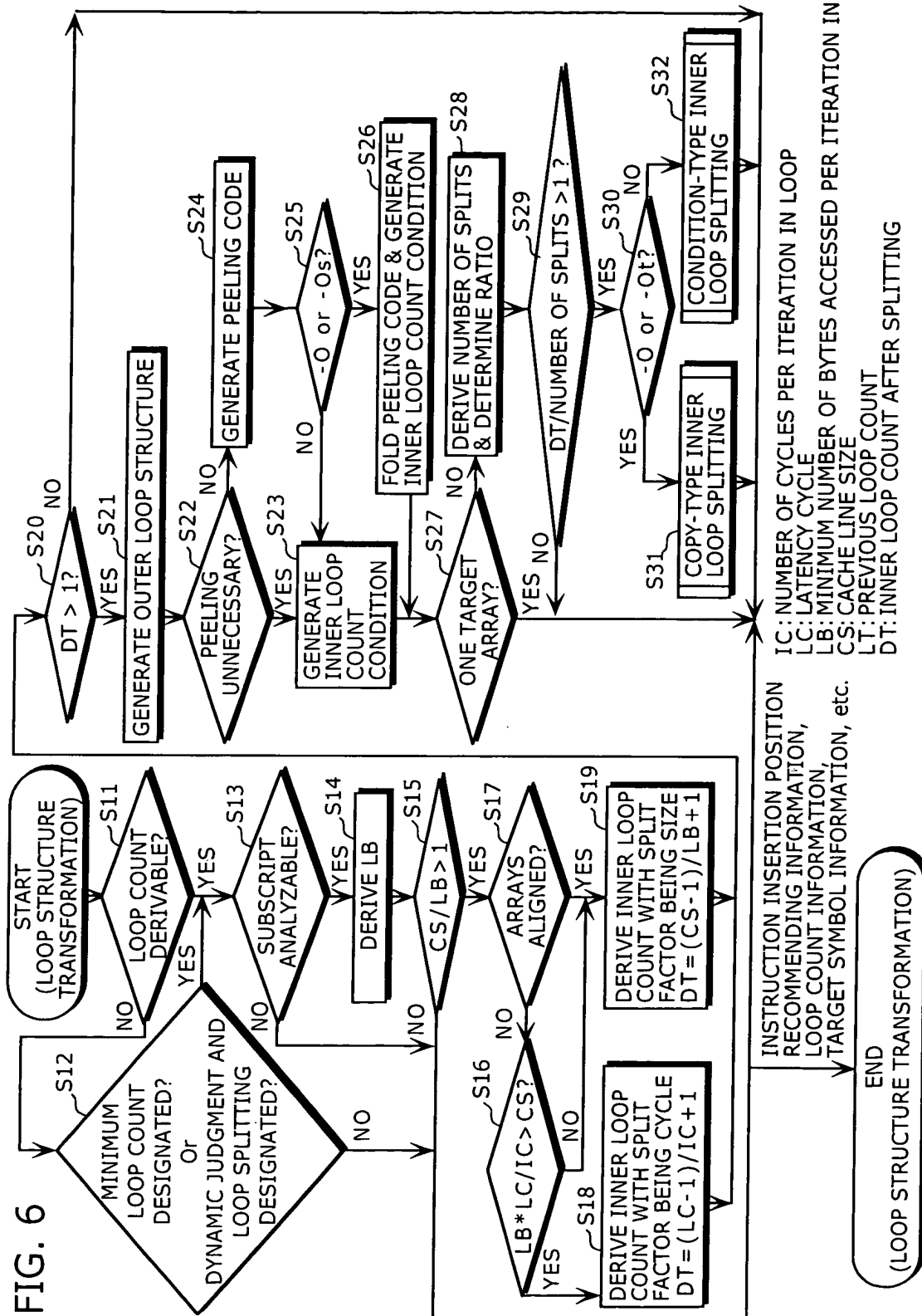


FIG. 7

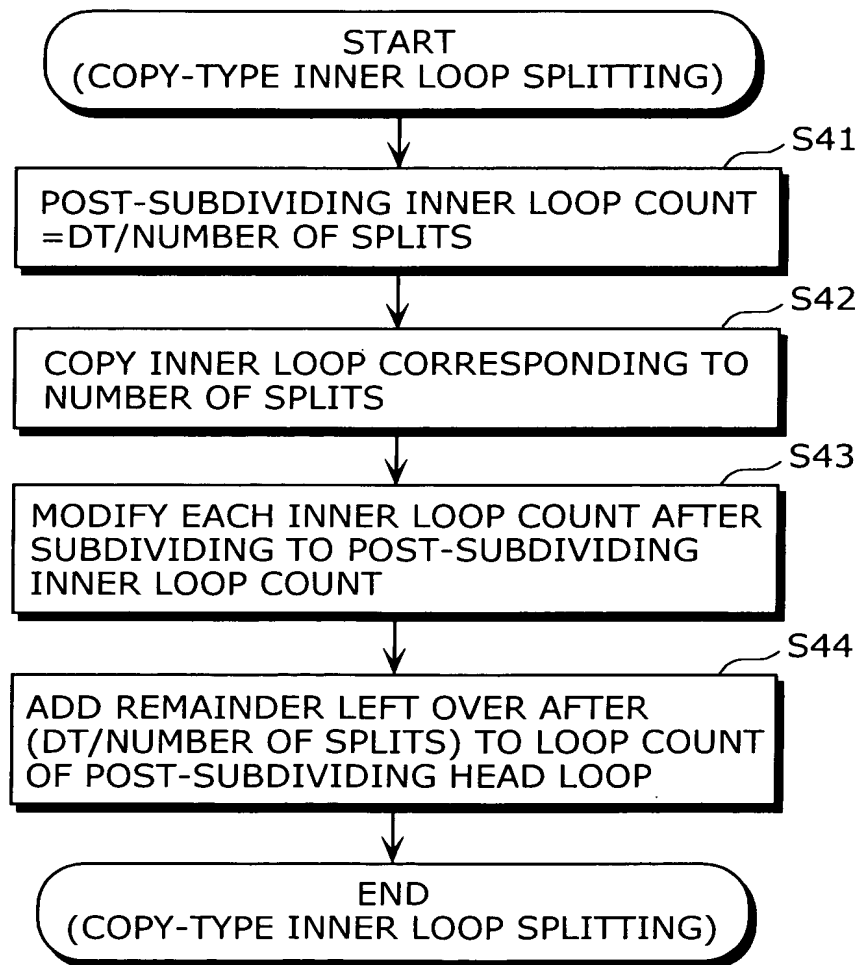


FIG. 8

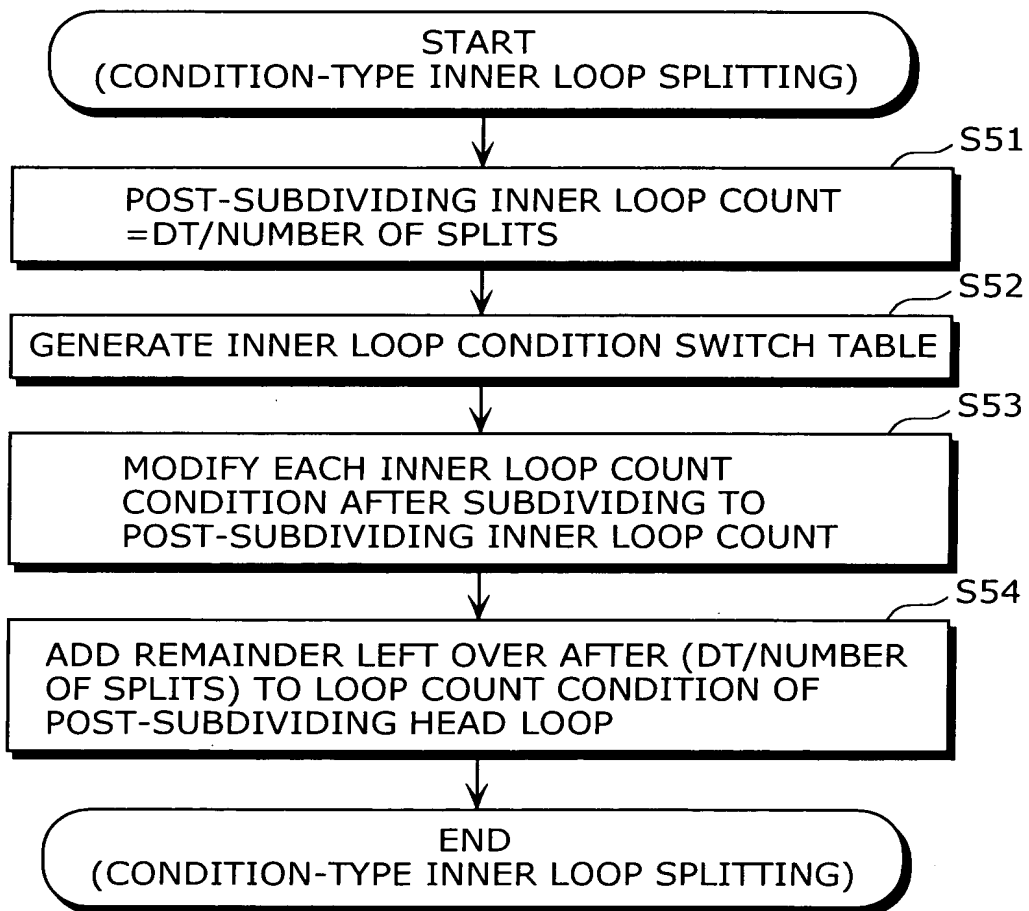


FIG. 9

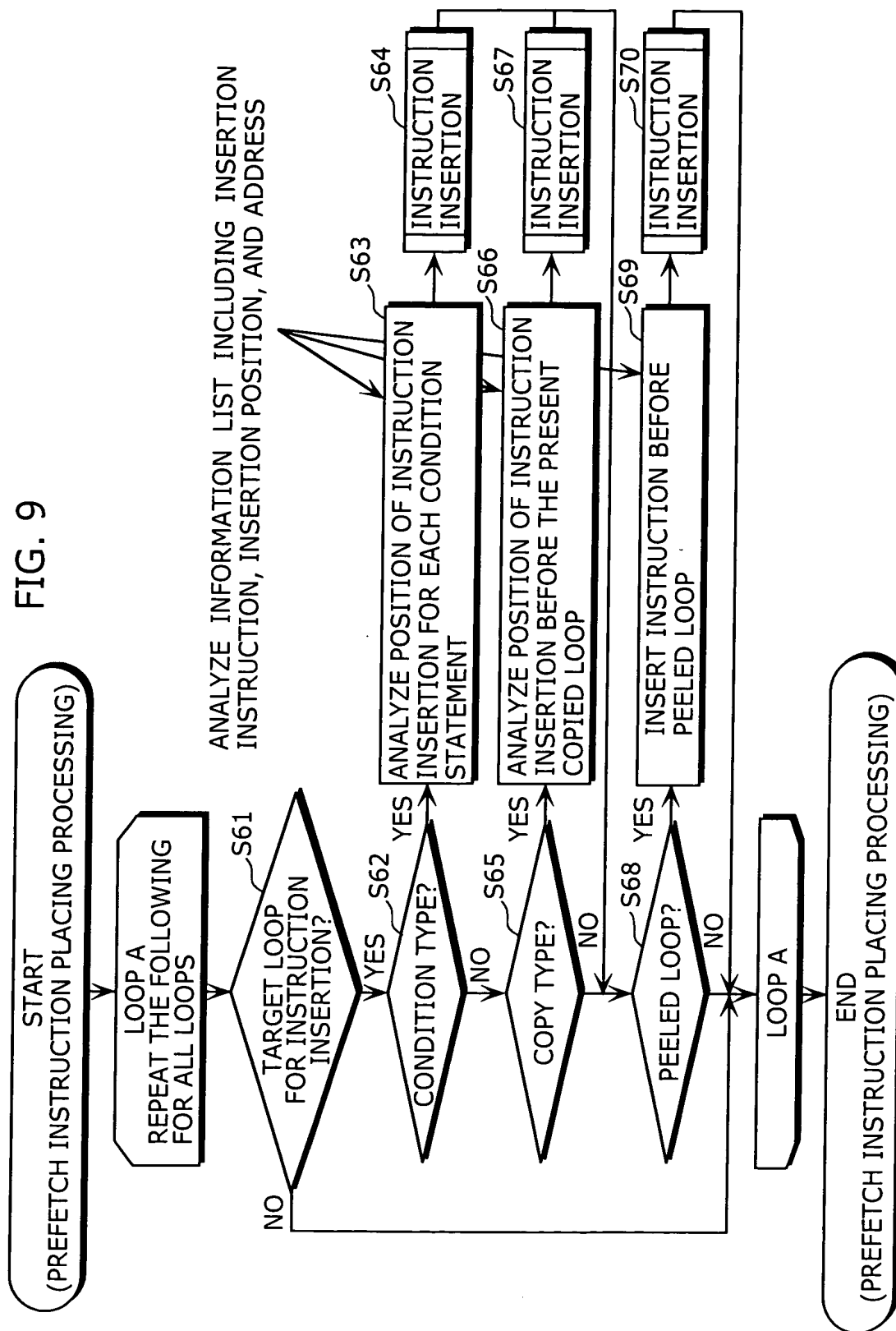


FIG. 10

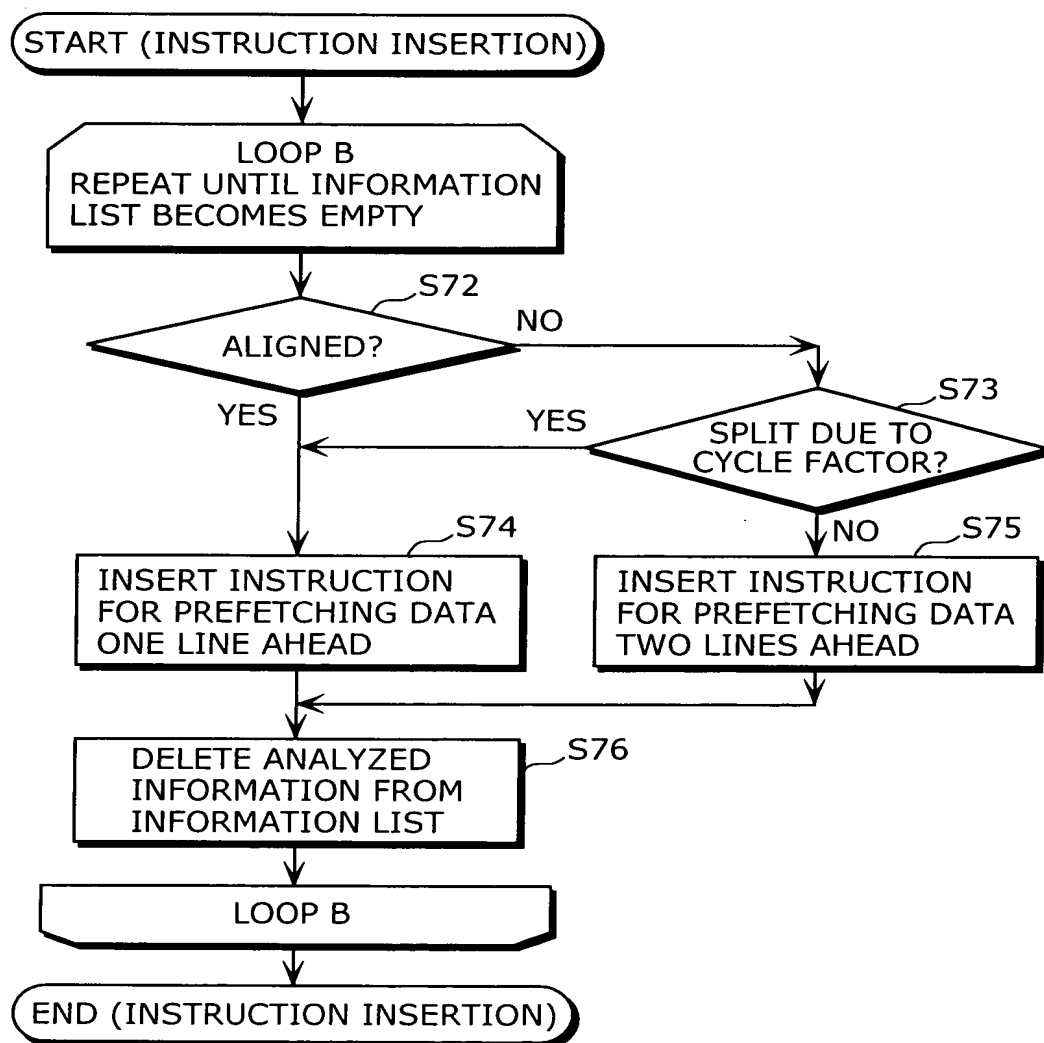


FIG. 11

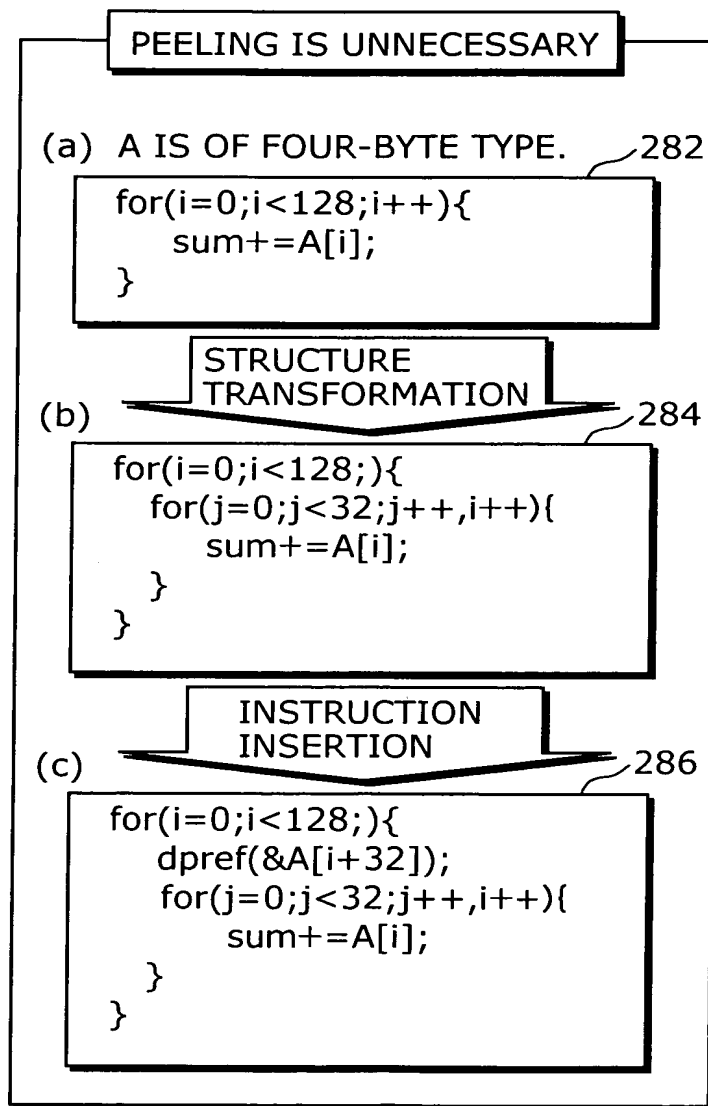


FIG. 12

INPUT PROGRAM SOURCE IN C LANGUAGE

```
int A[1000];  
  
int main(void)  
{  
    int i;  
    int sum = 0;  
  
    for ( i=0; j<128;i++) {  
        sum += A[ i ];  
    }  
  
    return sum;  
}
```

240

FIG. 13

INTERMEDIATE LANGUAGE INPUT BY TRANSFORMING UNIT

[PROLOG]		
[BGNBBLK] B1	[predecess set] no	[success set] B2
mov	REG (vr2) IMM(0)	
mov	REG (vr5) IMM(128)	
ld	REG (vr3) IMM(_A\$)	
mov	REG (vr1) IMM(vr2)	
[ENDBBLK]		
[BGNBBLK] B2	[predecess set] B1 B2	[success set] B2 B3
[label] L00001		
add	REG (vr2) REG (vr2), IMM(1)	
ldinc	REG (vr4), REG(vr3) INDIRECT(vr3,0), REG(vr3), IMM(4)	
cmplt	FLAG(C6) REG (vr2), REG (vr5)	
add	REG (vr1) REG (vr4), REG (vr1)	
jmpf	FLAG (C6), LAB(L00001)	
[ENDBBLK]		
[BGNBBLK] B3	[predecess set] B2	[success set] no
mov	REG (r0) REG (vr1)	
ret		
[ENDBBLK]		
[EPILOG]		

FIG. 14 INTERMEDIATE LANGUAGE AFTER TRANSFORMATION

```

[PROLOG]
[BGNBBLK] B1      [predecess set] no      [success set] B4
                   mov      REG (vr2) | IMM(0)
                   mov      REG (vr5) | IMM(32)
                   ld       REG (vr3) | MEM(_A$)
                   mov      REG (vr1) | REG(vr2)
                   mov      REG (vr7) | IMM(128)

[ENDBBLK]
[BGNBBLK] B4      [predecess set] B1 B4      [success set] B2
[label] L00002

[ENDBBLK]
[BGNBBLK] B2      [predecess set] B4 B2      [success set] B2 B5
[label] L00001
                   add      REG (vr2) | REG (vr2), IMM(1)
                   ldinc    REG (vr4), REG(vr3) | INDIRECT(vr3,0), REG(vr3), IMM(4)
                   cmplt    FLAG(C6) | REG (vr2), REG (vr5)
                   add      REG (vr1) | REG (vr4), REG (vr1)
                   jmpf     | FLAG (C6), LAB(L00001)

[ENDBBLK]
[BGNBBLK] B5      [predecess set] B2      [success set] B5 B3
                   cmplt    FLAG (C6) | REG (vr2), REG (vr7)
                   jmpf     | FLAG (C6), LAB(L00002)

[ENDBBLK]
[BGNBBLK] B3      [predecess set] B5      [success set] no
                   mov      REG (r0) | REG (vr1)
                   ret

[ENDBBLK]
[EPILOG]

```

FIG. 15

INTERMEDIATE LANGUAGE AFTER INSTRUCTION INSERTION

270

```

[PROLOG]
[BGNBBLK] B1      [predecessor set] no      [success set] B4
                   mov      REG (vr2) | IMM(0)
                   mov      REG (vr5) | IMM(32)
                   ld       REG (vr3) | MEM(_A$)
                   mov      REG (vr1) | REG(vr2)
                   mov      REG (vr7) | IMM(128)

[ENDBBLK]
[BGNBBLK] B4      [predecessor set] B1 B4      [success set] B2
[label] L00002

[ENDBBLK]
[BGNBBLK] B2      [predecessor set] B4 B2      [success set] B2 B5
[label] L00001

                   dpref      | INDIRECT(vr2, 128), REG (vr2)

                   [predecessor set] B4 B2      [success set] B2 B5

                   add      REG (vr2) | REG (vr2), IMM(1)
                   ldinc    REG (vr4), REG(vr3) | INDIRECT(vr3,0), REG(vr3), IMM(4)
                   cmplt    FLAG(C6) | REG (vr2), REG (vr5)
                   add      REG (vr1) | REG (vr4), REG (vr1)
                   jmpf     | FLAG (C6), LAB(L00001)

[ENDBBLK]
[BGNBBLK] B5      [predecessor set] B2      [success set] B5 B3
                   cmplt    FLAG (C6) | REG (vr2), REG (vr7)
                   jmpf     | FLAG (C6), LAB(L00002)

[ENDBBLK]
[BGNBBLK] B3      [predecessor set] B5      [success set] no
                   mov      REG (r0) | REG (vr1)
                   ret

[ENDBBLK]
[EPILOG]

```

FIG. 16

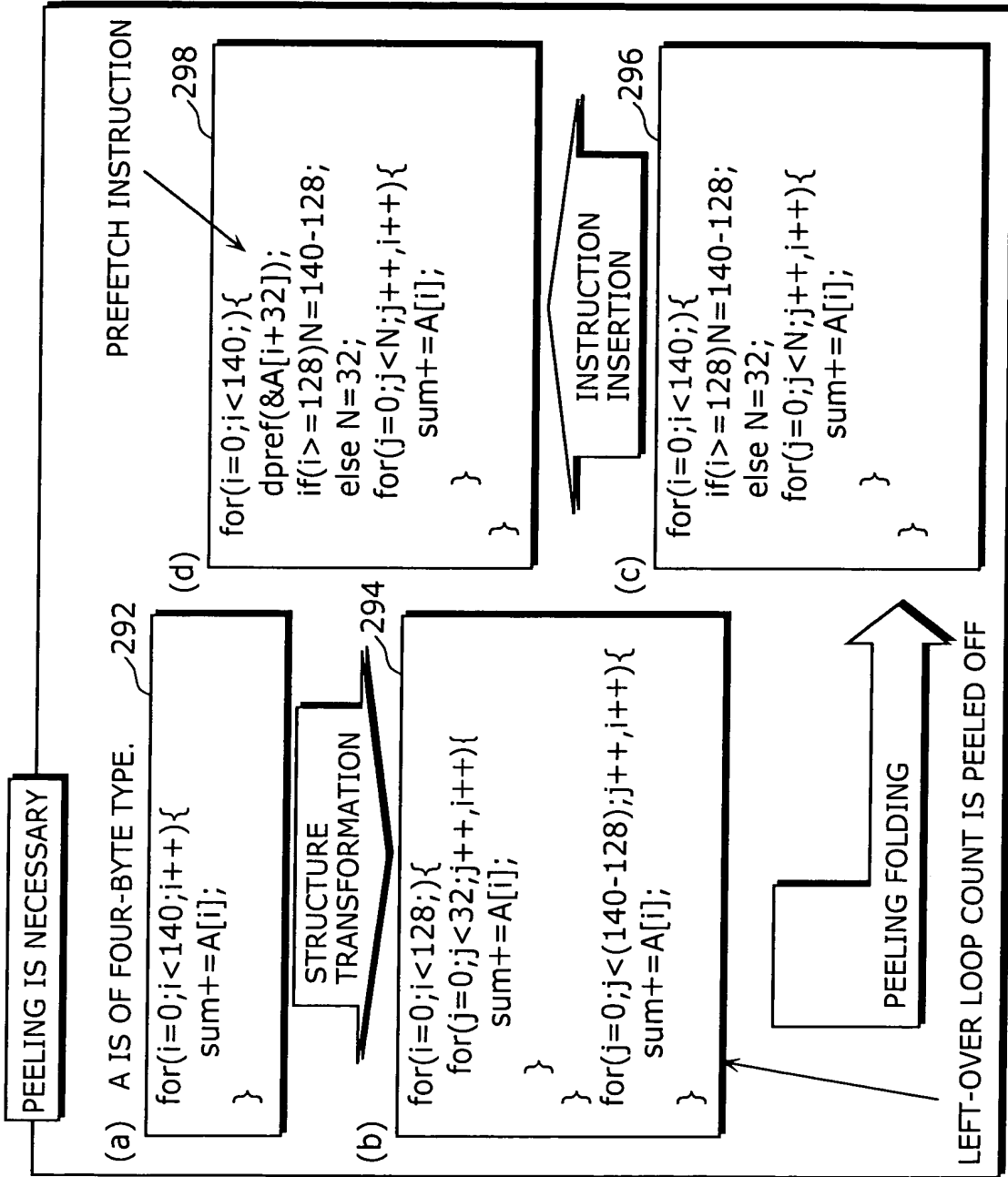


FIG. 17

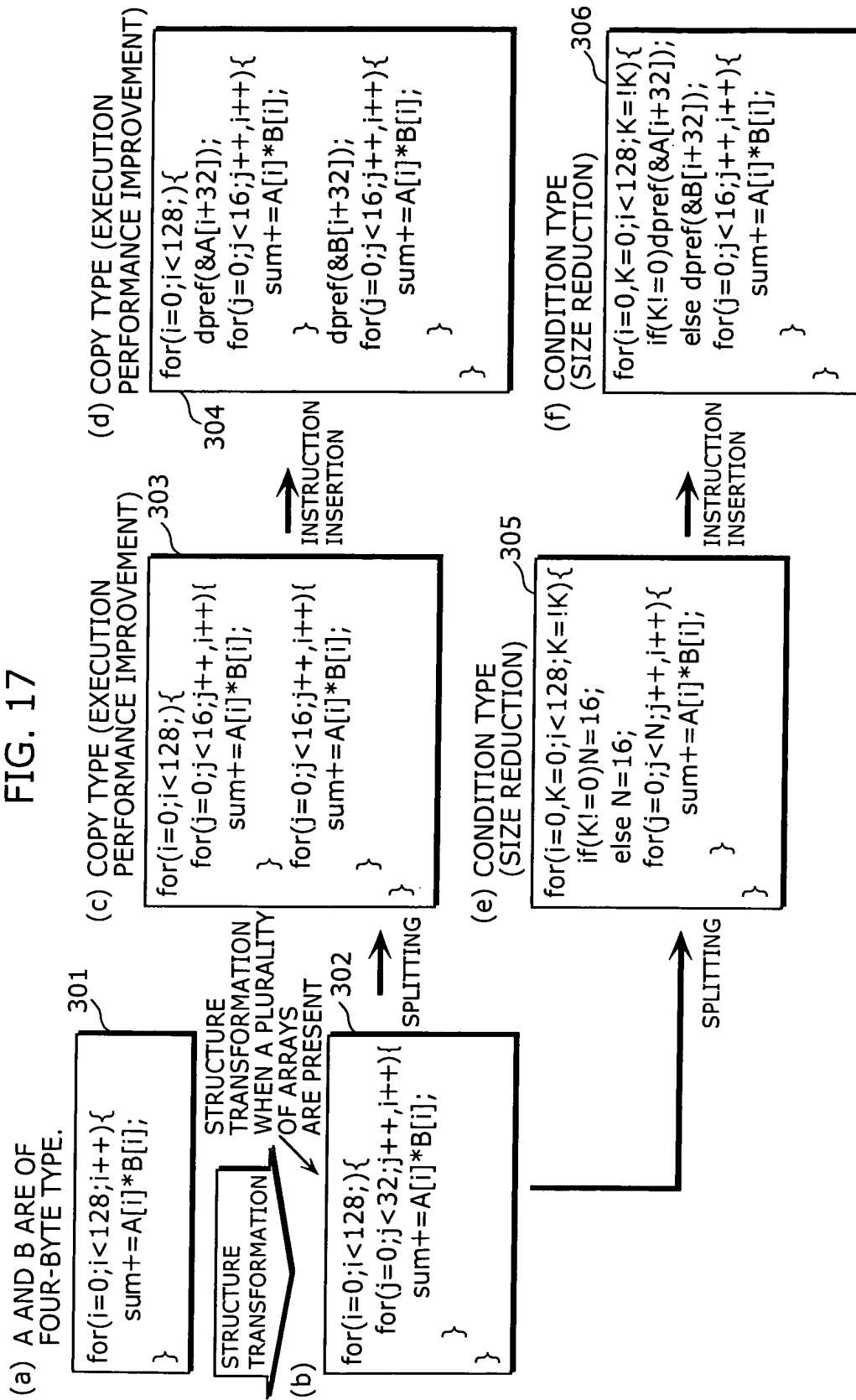


FIG. 18

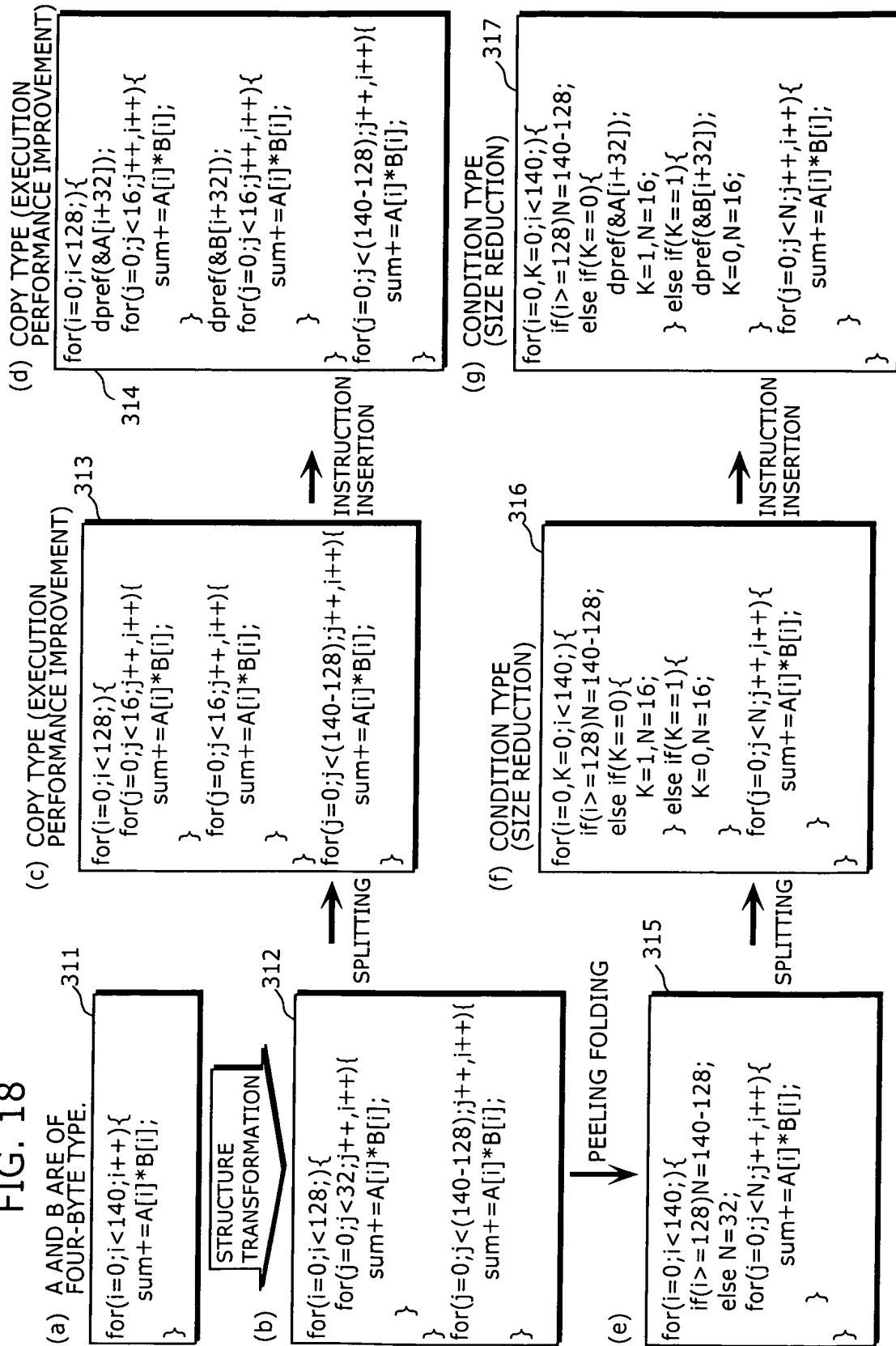
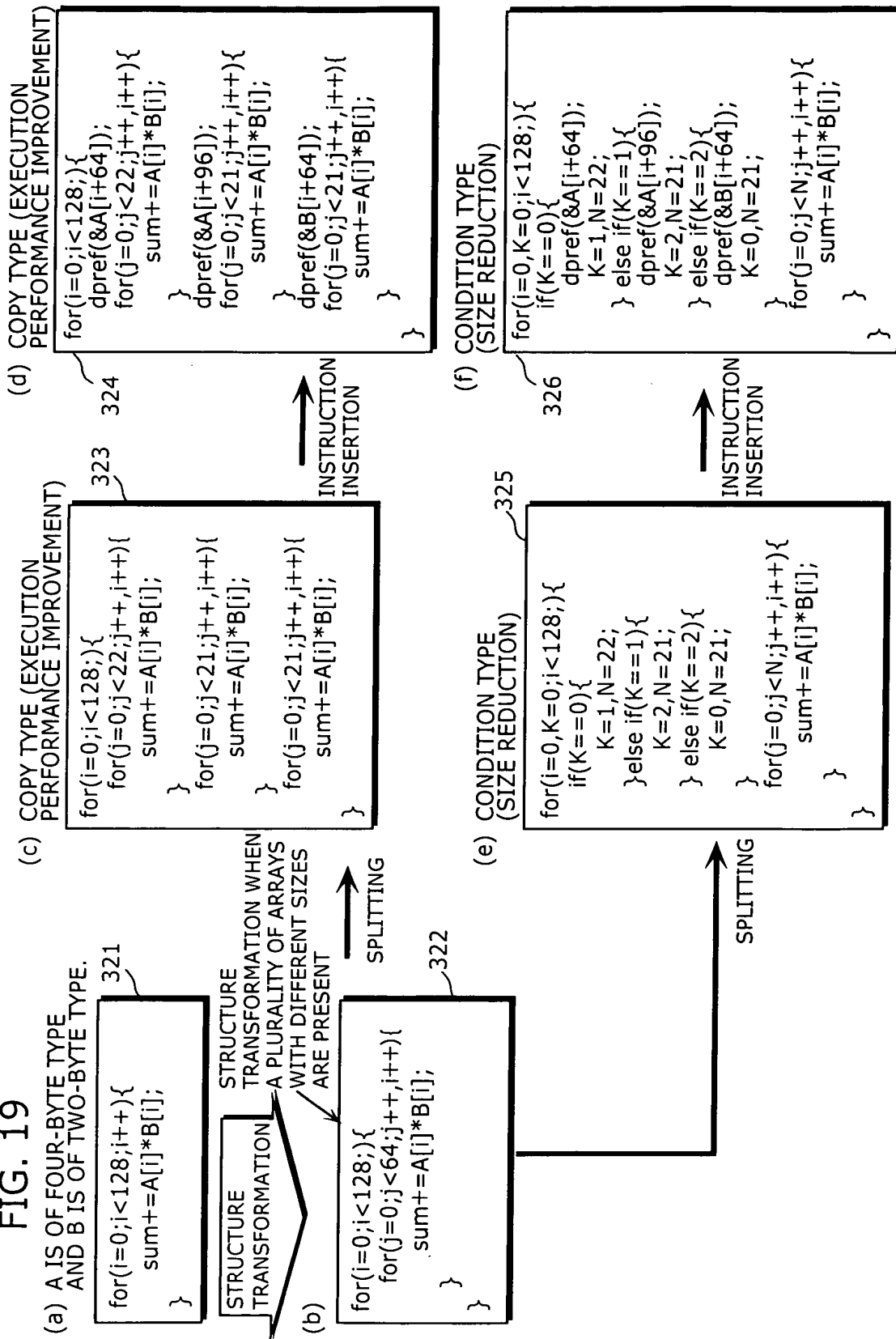


FIG. 19



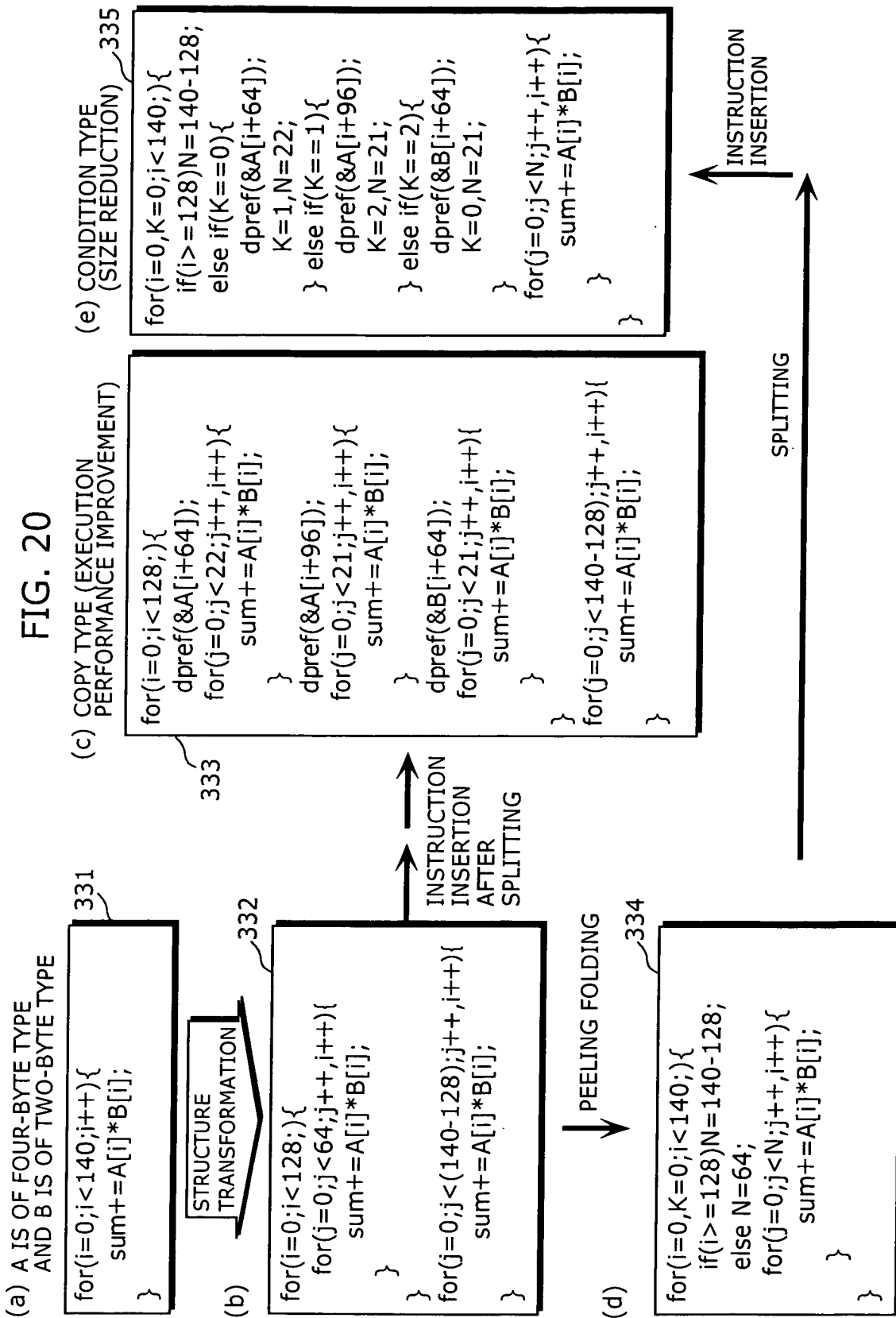


FIG. 21

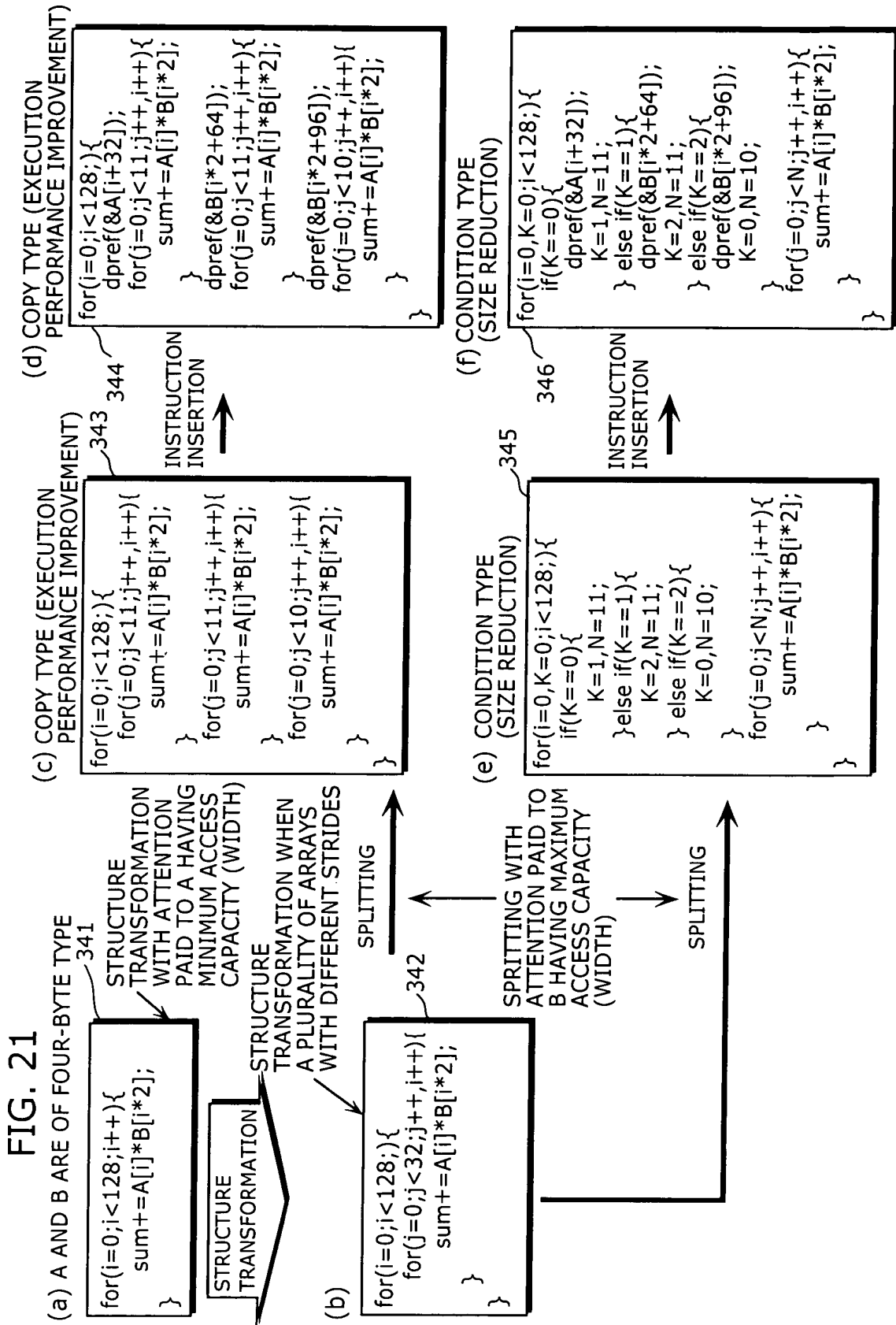


FIG. 22

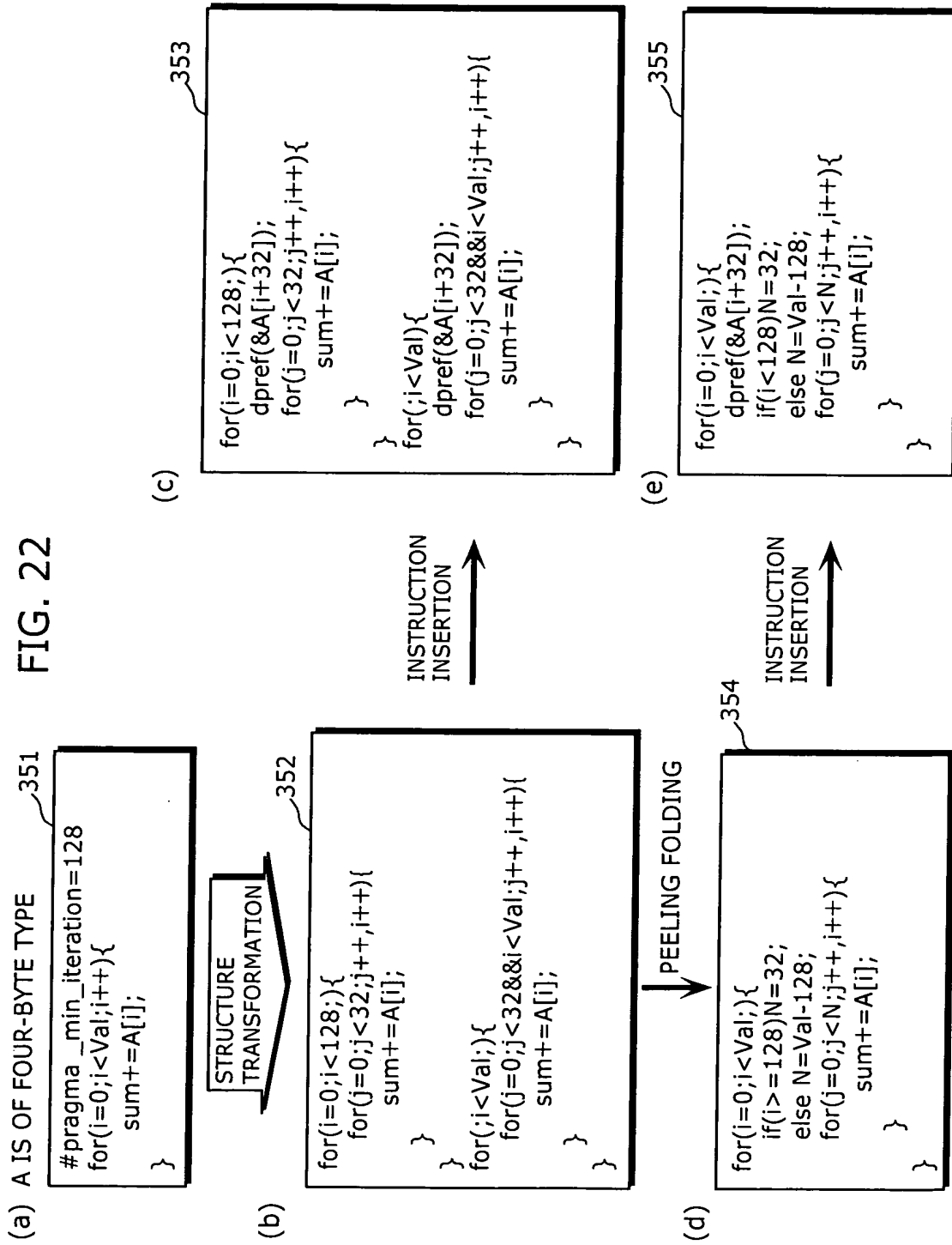


FIG. 23

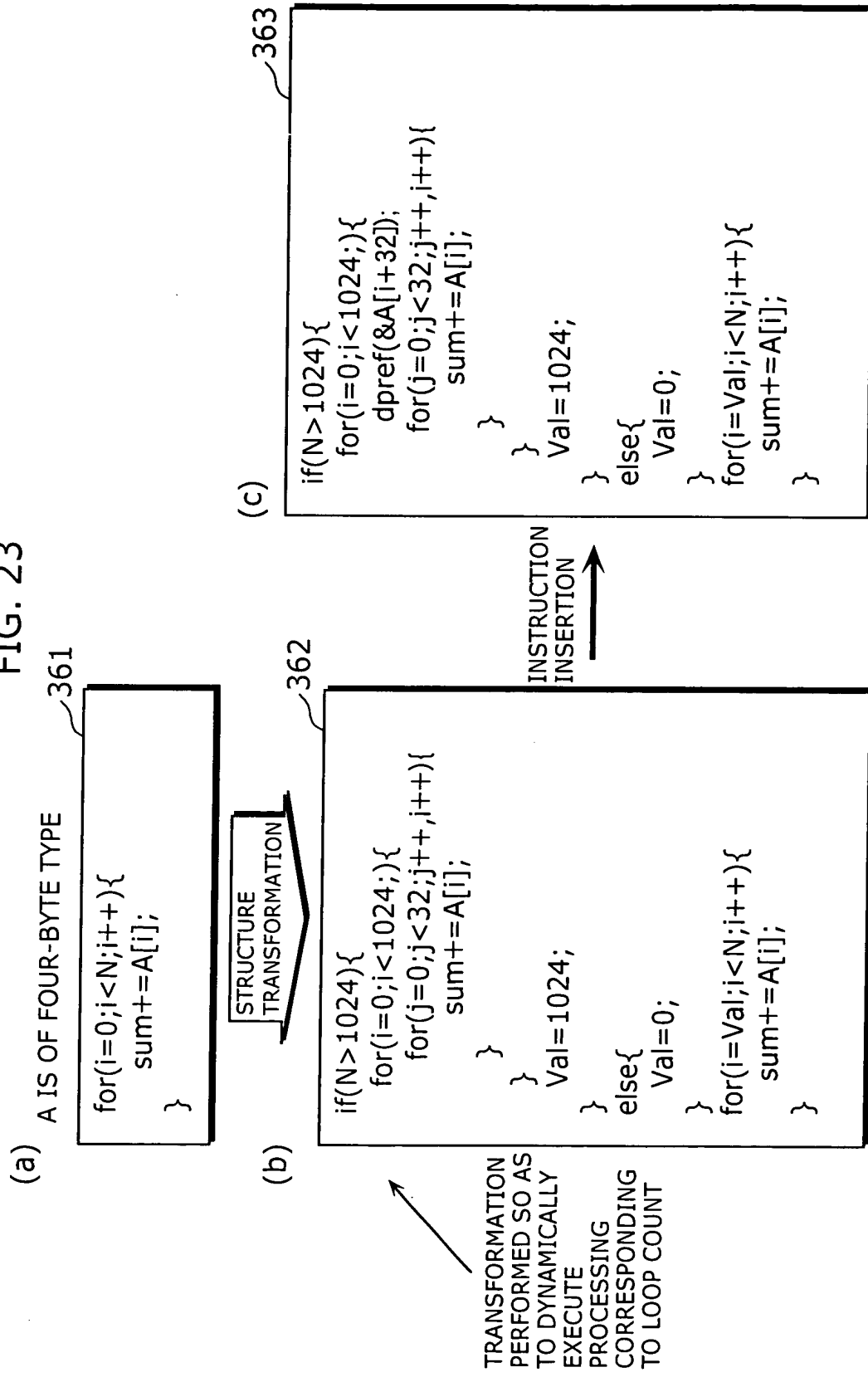


FIG. 24

(a) A IS OF FOUR-BYTE TYPE

371

```
for(i=0;i<N;i++){  
    sum+=A[i];  
    sum+=A[i+1];  
    sum+=A[i+2];  
    ~SKIP~  
    sum+=A[i+30];  
    sum+=A[i+31];  
}
```

WHEN IT IS JUDGED
THAT LOOP STRUCTURE
TRANSFORMATION
IS UNNECESSARY,
INSTRUCTION IS
INSERTED WITHOUT
STRUCTURE
TRANSFORMATION.

INSTRUCTION
INSERTION

(b)

372

```
for(i=0;i<N;i++){  
    dpref(&A[i+32]);  
    sum+=A[i];  
    sum+=A[i+1];  
    sum+=A[i+2];  
    ~SKIP~  
    sum+=A[i+30];  
    sum+=A[i+31];  
}
```


FIG. 25

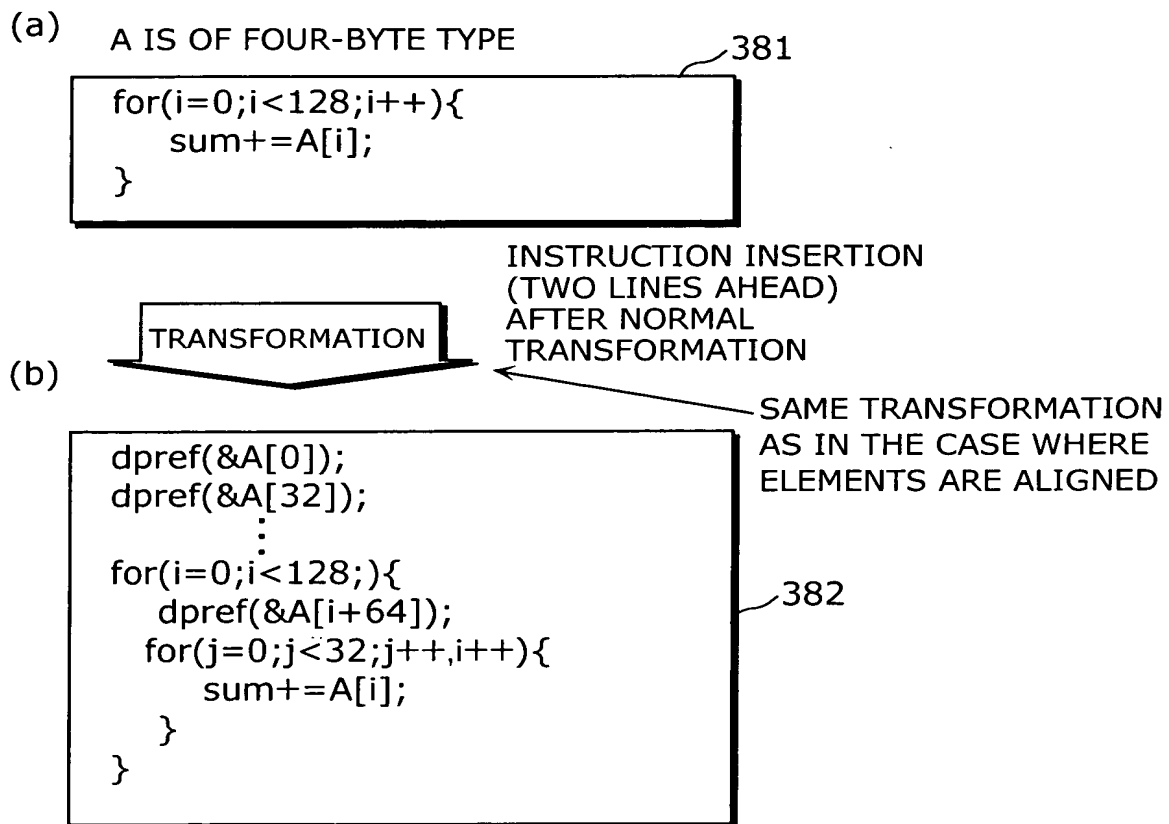


FIG. 26

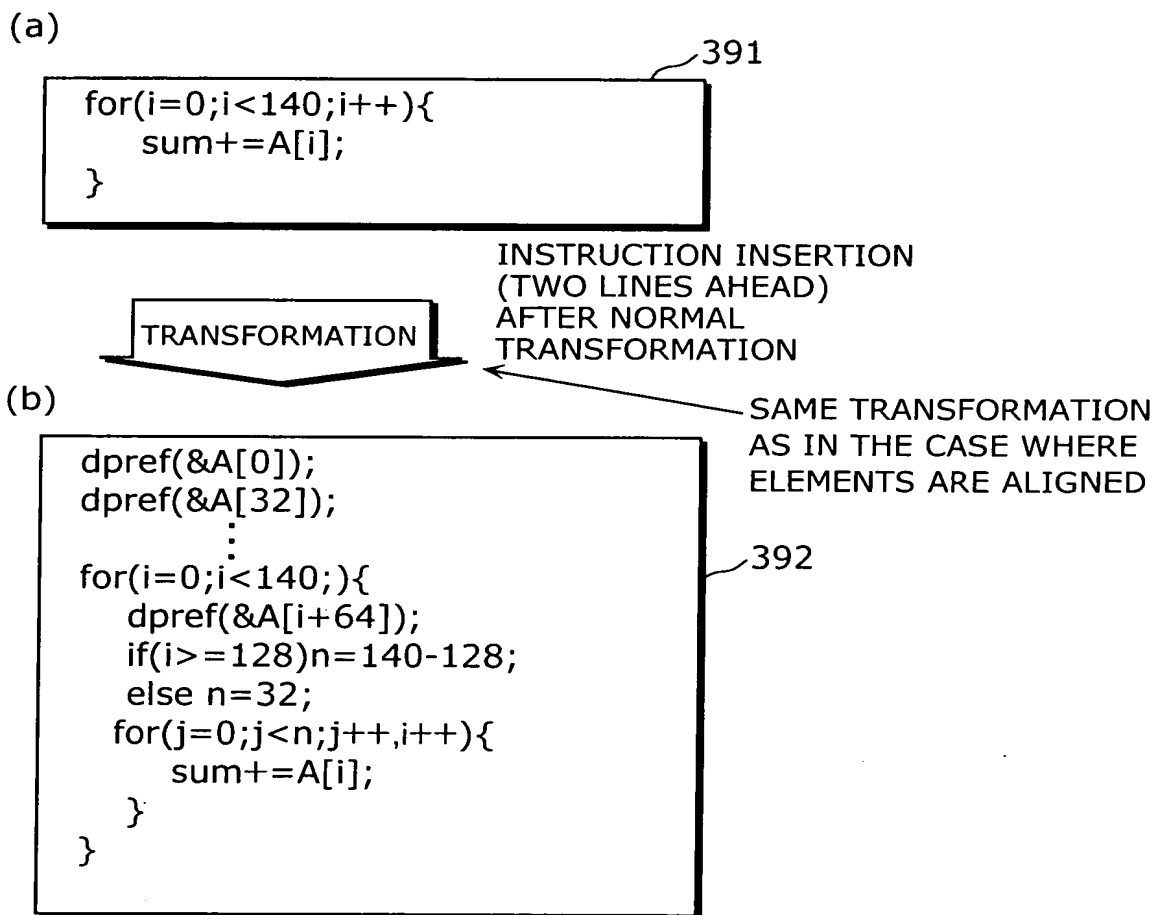


FIG. 27

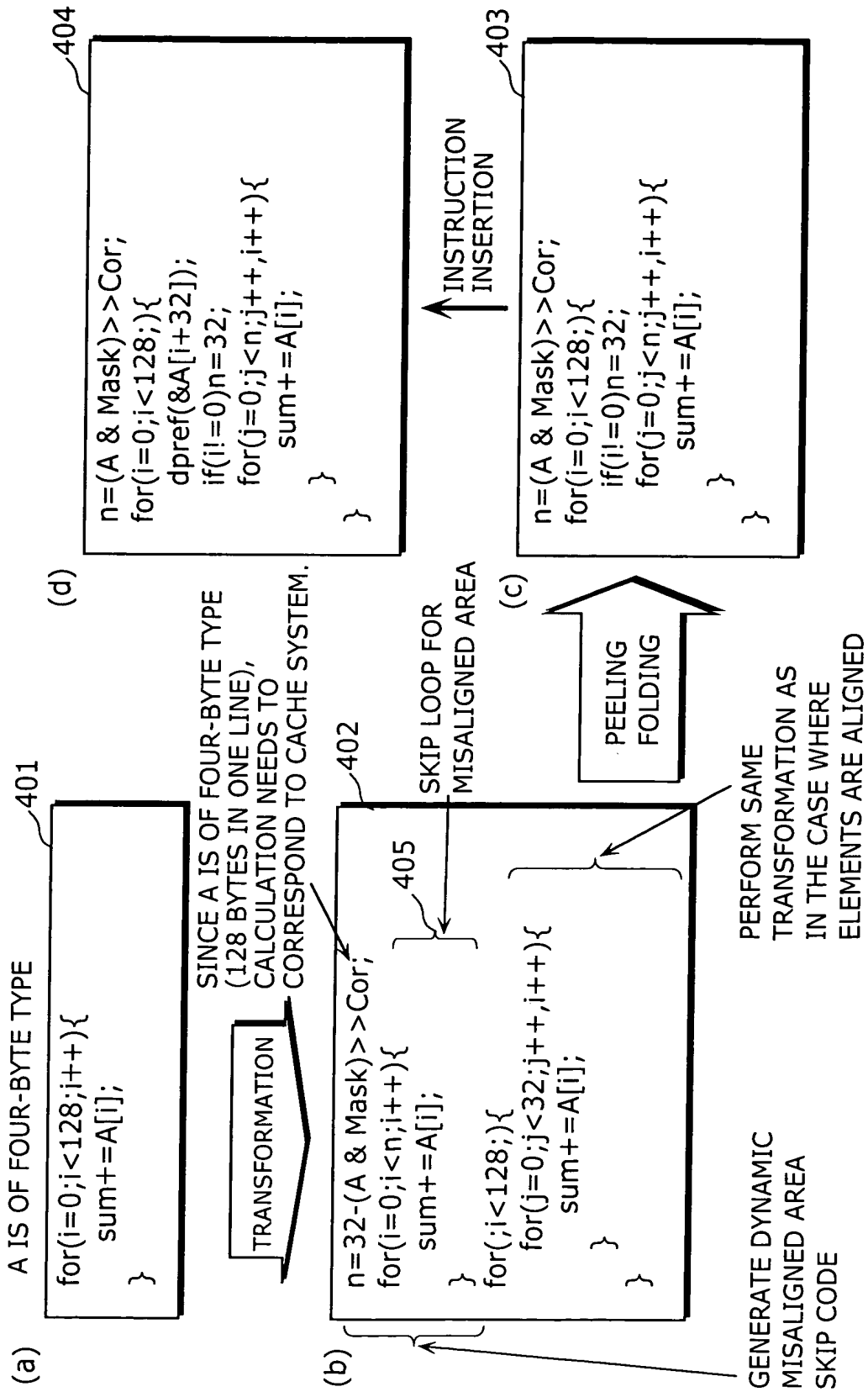


FIG. 28

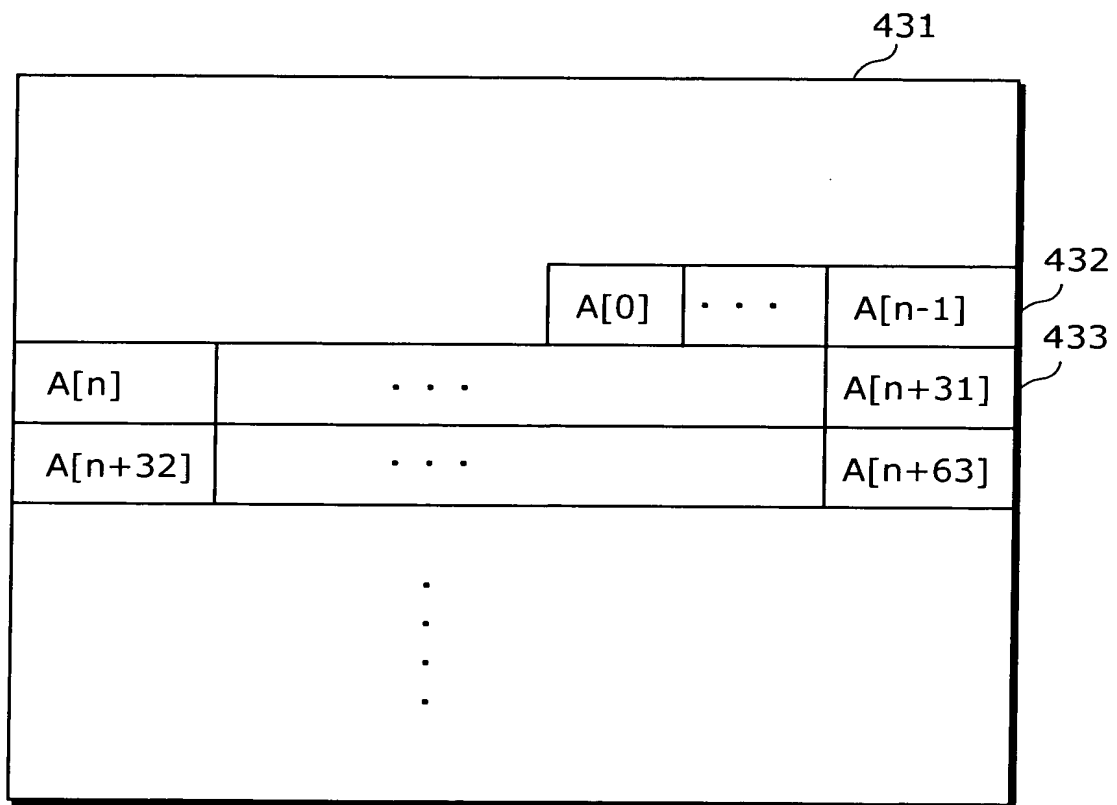


FIG. 29

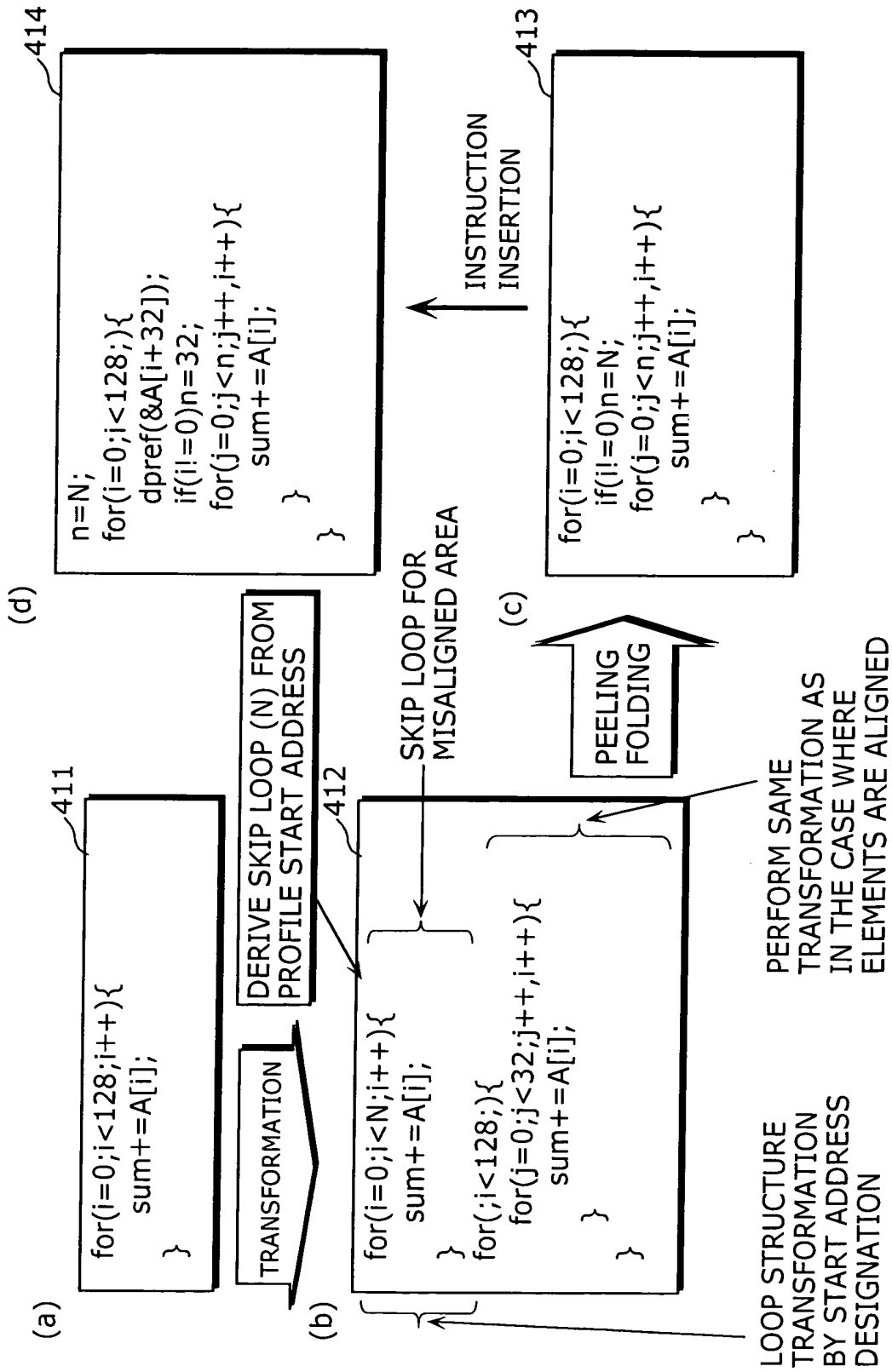


FIG. 30

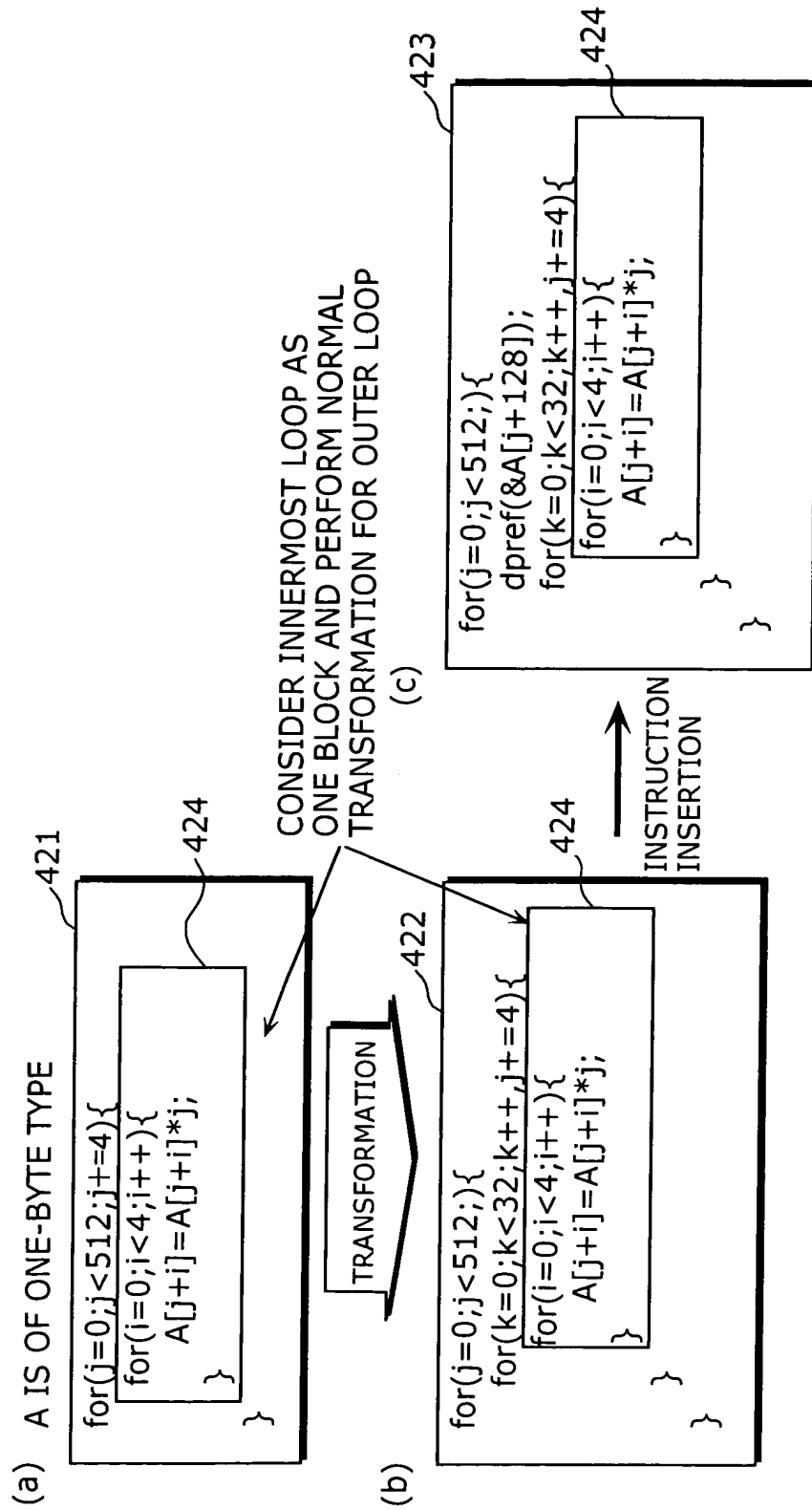


FIG. 31

(a)

```
int b[128]:  
#pragma _loop_tiling_dpref b  
for (i=0; i<128; i++)  
{  
    a[i] = b[i];  
}
```

(b)

```
for (i=0; i<128; )  
{  
    dpref(&b[i+32]);  
    for (j =0; j<32; j++, i++){  
        a[i] = b[i];  
    }  
}
```

FIG. 32

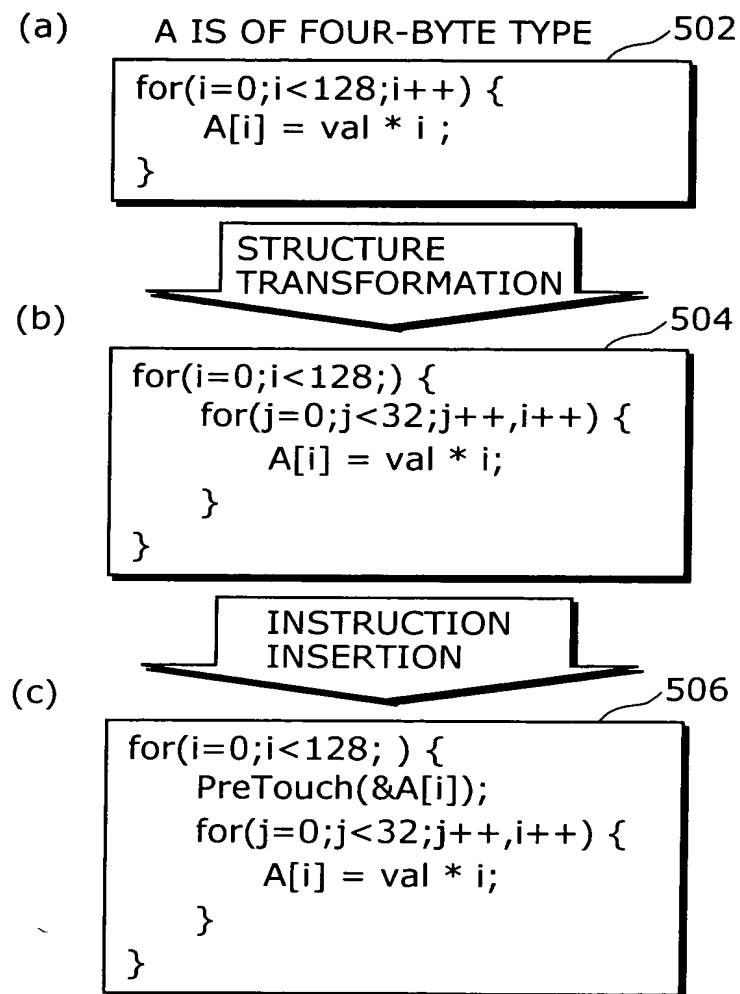


FIG. 33

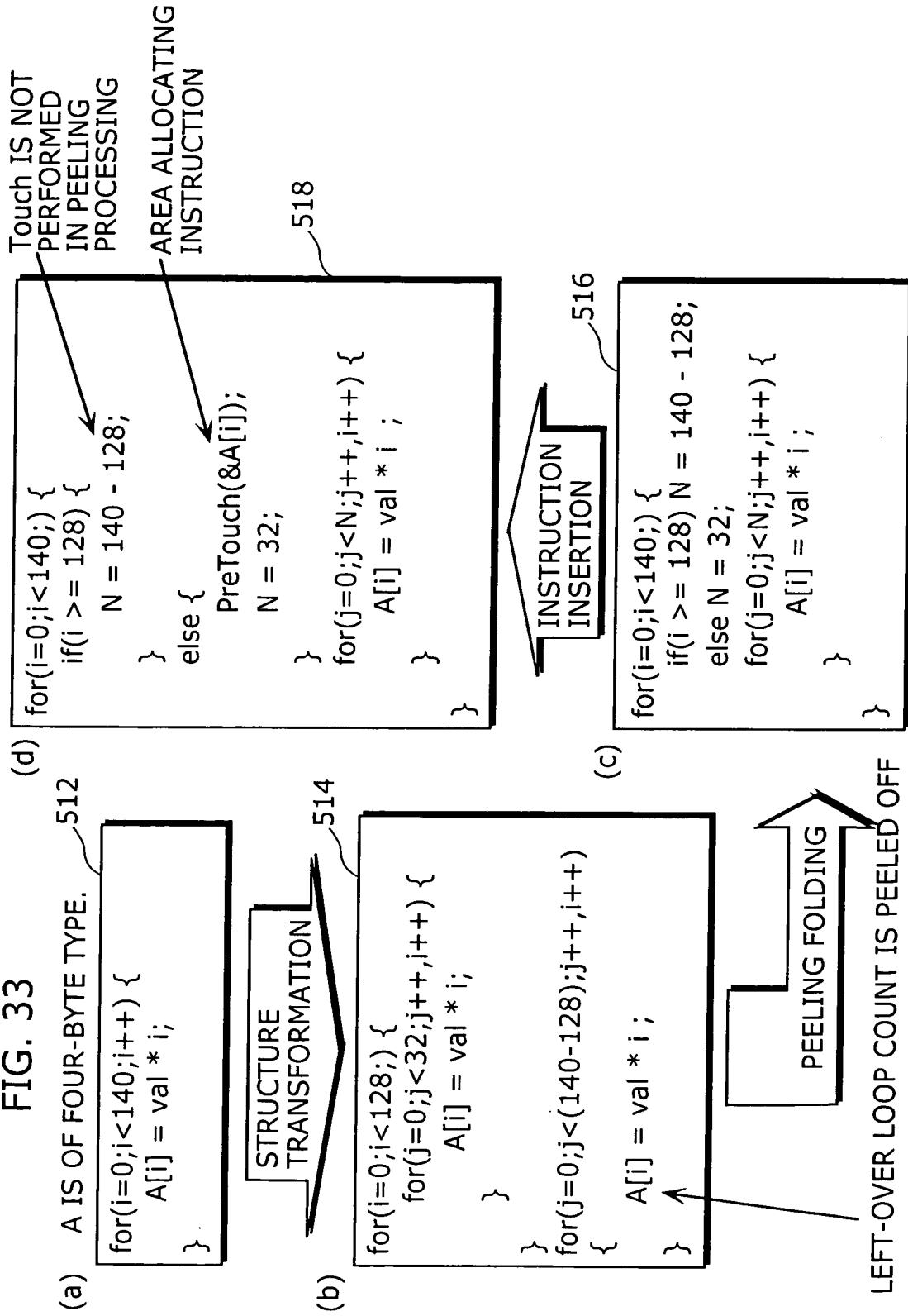


FIG. 34

